PATTERSON & SHERIDAN, LLP
ATTORNEYS AT LAW

3040 Post Oak Blvd, Suite 1500
Houston, TX 77056-6582
TEL 713.623.4844
FAX 713.623.4846

RECEIVED
CENTRAL FAX CENTER
FEB 0 6 2006

FACSIMILE COVER SHEET

DATE: February 6, 2006

FILE NO: ROC920010193US3 (IBMK10195)

TO: MAIL STOP APPEAL BRIEF – PATENTS
Examiner David R. Lazaro

FAX NO: 1-571-273-8300

FROM: Gero G. McClellan/Jon Stewart

PAGE(S) with cover: 21

RE:

TITLE: METHOD FOR EFFICIENTLY CONTROLLING SOCKET SERVER SEND BUFFER USAGE

U.S. SERIAL NO.: 10/037,595
FILING DATE: January 4, 2002
INVENTOR(S): Michael Edward Baskey et al.
EXAMINER: David R. Lazaro
GROUP ART UNIT: 2155
CONFIRMATION NO.: 6369

Attached are the following document(s) for the above-referenced application:

Appeal Brief.

435304_1.DOC

PATENT
Atty. Dkt. No. ROC920010193US3
PS Ref. No.: IBMK10195

RECEIVED
CENTRAL FAX CENTER
FEB 0 6 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Baskey et al.	§	Confirmation No.: 6369
Serial No.: 10/037,595	§	Group Art Unit: 2155
Filed: January 4, 2002	§	Examiner: David R. Lazaro
For: METHOD FOR EFFICIENTLY CONTROLLING SOCKET SERVER SEND BUFFER USAGE	§	

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner David R. Lazaro, on the date shown below:

February 6, 2006
Date

Jon Stewart

Dear Sir:

APPEAL BRIEF

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2155 dated September 7, 2005, finally rejecting claims 1-3, 5-10, 12, 13 and 15-34. The final rejection of claims 1-3, 5-10, 12, 13 and 15-34 is appealed. This Appeal Brief is believed to be timely since facsimile transmitted by the due date of February 6, 2006, as set by mailing a Notice of Appeal on December 6, 2005. Please charge the fee of $500.00 for filing this brief to Deposit Account No. 09-0465/ROC920010193US3.

02/08/2006 WABDELR1 00000039 090465 10037595
01 FC:1402 500.00 DA

PATENT
Atty. Dkt. No. ROC920010193US3
PS Ref. No.: IBMK10195

TABLE OF CONTENTS

PATENT
Atty. Dkt. No. ROC920010193US3
PS Ref. No.: IBMK10195

Real Party in Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicant asserts that no other appeals or interferences are known to the Applicant, the Applicant's legal representative or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1-3, 5-10, 12, 13 and 15-34 are pending in the application. Claims 1-34 were originally presented in the application. Claims 4, 11 and 14 have been canceled without prejudice. Claims 1-3, 5-10, 12, 13 and 15-34 stand finally rejected as discussed below. The final rejections of claims 1-3, 5-10, 12, 13 and 15-34 are appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

All claim amendments have been entered by the Examiner, including amendments to the claims proposed after the final rejection.

Summary of Claimed Subject Matter

Claimed embodiments include a method (see e.g., claim 1), a computer readable medium containing an application program (see e.g., claim 12), and a computer in a distributed environment (see e.g., claim 24) for a server application to process messages from a client application. See e.g., *Application*, ¶ 1, 13, Abstract.

Claimed embodiments include a method (see e.g., claims 1-3, 5-10) of processing messages in a computer. See e.g., Application, ¶ 1, 13, 14. The claimed embodiment includes, in response to a request from a server application, allocating a system-supplied buffer to the server application. See e.g., *Application* ¶ 44, 45, 78. The server application is configured to exchange data with a client application running on another computer using a network-based socket. See e.g., *Application*, ¶ 48, 82, Figure 4. The system supplied buffer allocated to the server application is of a sufficient size to contain the data. See e.g., *Application*, ¶ 64, 76, 84. The method includes writing the data to the system-supplied buffer. See e.g., *Application*, ¶ 78, 84, 85, Figures 12, 13. Once the data is written to the system-supplied buffer, the method includes passing the system-supplied buffer to the network-based socket to allow the server application to continue processing while the data is sent to the client. See e.g., *Application*, ¶ 80, 86. The method also includes sending, by way of the network-based socket, the data from the system-supplied buffer to the other computer via a network, see e.g., *Application*, ¶ 78, 79, 85, Figures 12, 13, and freeing memory consumed by the system supplied buffer, See e.g., *Application*, ¶ 80 and 86.

Claimed embodiments also include a computer readable medium containing an application program (see e.g., claims 12, 13, 15-23) configured to perform operations for processing messages in a computer. See *Application*, ¶ 1, 13, 15, 35, 75.

This claimed embodiment includes, in response to a request from a server application, allocating a system-supplied buffer to the server application. See e.g., *Application* ¶ 44, 45, 78. The server application is configured to exchange data with a client application running on another computer using the communications program. See e.g., *Application*, ¶ 48, 82, Figure 4. Also, the system supplied buffer is of a sufficient size to contain the data. See e.g., *Application*, ¶ 64, 76, 84. The operations performed

by the program include receiving the system-supplied buffer from the server application, wherein the system-supplied buffer contains data written to the system-supplied buffer by the server application. See e.g., *Application*, ¶ 76, 77, 83, 84, Figures 12, 13. The operations also include sending, by way of the communications program, the data from the system-supplied buffer to the other computer via a network, thereby allowing the server application to continue processing while the data is sent to the client. See e.g., *Application*, ¶ 78, 79, 80, 85 86, Figures 12, 13. Thereafter, the operations include returning the allocated system supplied buffer to the computer. See e.g., *Application*, ¶ 80, 86.

Claimed embodiments (see e.g., claims 24-34) also include a computer in a distributed environment. See e.g., *Application*, ¶ 1, 13, 15. The computer includes a network interface configured to support a network connection with at least one other computer in the distributed environment. See e.g., Application ¶ 37-41, Figure 1. The computer includes a memory (see e.g., *Application*, ¶ 42-43, Figure 1) containing contents. The contents of the memory include an operating system and a server application, a sockets-based communication facility. See e.g., *Application*, ¶ 43-48.

The memory also includes a system-owned memory space from which to allocate system-supplied buffers. See e.g., *Application*, 43, 44, 76-79, and 81-83. The memory of the computer further includes an application-owned memory space owned by the server application. See e.g., *Application*, 79, 80, 84. The computer also includes a processor configured by at least a portion of the contents to perform operations for processing client-server messages. See e.g., *Application*, 15, 40-42. The operations include, in response to a request from the server application, allocating a system-supplied buffer to the server application. See e.g., *Application* ¶ 44, 45, 78. The server application is configured to exchange data with a client application running on another computer using a network-based socket. See e.g., *Application*, ¶ 80, 86, Figures 12, 13. The system supplied buffer is of a sufficient size to contain the data. See e.g., *Application*, ¶ 64, 76, 84.

Grounds of Rejection to be Reviewed on Appeal

1. Claims 1-3, 5-10,12-13,15-21 and 24-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent Application Publication 2003/0217184 by *Nair* in view of U.S. Patent 6,055,576 by *Beighe*.

ARGUMENTS

Obviousness of Claims 1-3, 5-10,12-13,15-21 and 24-31 over *Nair* in view of *Beighe*

The Applicable Law

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the first and third criteria.

The References

Nair discloses a method of enhancing a data communications pathway by maintaining a pool of buffers managed by a buffer manager. "With reference to FIG. 3, the buffer manager 114 maintains a pool of available buffers from which a protocol module may select or be allocated a buffer for temporary storage of the frame of data." *Nair*, ¶ 25. *Nair* further discloses that protocol modules may share a pointer to a buffer across different layers of a protocol stack (e.g., passing a pointer from a physical layer, to a data link layer, to a network layer, etc.). Thus, *Nair* discloses maintaining the data frames in a common buffer space, referenced by a software module at each layer by the shared pointer. A passage from one of the paragraphs cited by the Examiner confirms this:

> In particular, when a frame or cell of data is received from a network attached to the machine, or when a packet of data is prepared for transmission over a network attached to the machine, as the data is processed by each appropriate protocol software module, the data is maintained in the same buffer space. Only the pointers to the data space need be passed between the protocol software modules so that the protocol software modules that process the data know where to access the data.

Nair, ¶ 20. Importantly, *Nair* teaches that the shared buffer used by the protocol stack may be discarded (or returned to the buffer pool) once a frame is provided to a server application. Specifically, *Nair* provides:

> "[P]rocessing of the data frame continues up the protocol stack until processing of the data frame by the machine is competed. **At such time, the data is read from the buffer at 230 and, for example, provided to an application software program.** At this point, for example, **the buffer is no longer needed** for temporarily storing the data pockets while the various protocol software modules in the protocol stack process the data frame."

Nair, ¶ 28.

Nair does disclose that network communications may be bi-directional, and that when a data frame is recited by a protocol module for transmission, the protocol module may allocate a buffer from the "buffer pool." Nevertheless, buffers from the "buffer pool" are allocated by a protocol module, only <u>after</u> receiving a data frame from a server application. Specifically, *Nair* provides: "It is appreciated that the process of the present invention is equally applicable to receiving at the top of the protocol stack a data frame from a higher layer application program, and passing control of processing the frame of data down the protocol stack in the machine in preparation for transmitting the data frame from the machine and over the attached network to another machine connected to the network." *Nair*, ¶ 30. In other words, *Nair* discloses that the use of the "buffer pool" is localized to the protocol modules, independent from the operation of any higher level application programs.

Beighe discloses a method for accepting or rejecting a data packet which is being transferred between a client and server over a cable and a cable communication network. See *Beighe*, Abstract. Unlike a dedicated connection established using a telephone modem, a cable television network includes many nodes that each hear any communication over the physical cable. *Beighe* discloses a technique for a cable modem to selectively "listen" to data transmitted over the cable network. See *Beighe*, 1:52-67. Like the description of the TCP protocol in *Nair* at paragraph 2, *Beighe* discloses that data communications may occur using a TCP/IP based collection of

network modules. Specifically, the Examiner cites the following description of the TCP protocol, as used in a cable-television based network:

> Applications programs 30 provide the user interface. An applications interface layer 32, sometimes called sockets allows applications programs to communicate using a communications protocol such as TCP/IP layer 34. Applications interface 32 insures interoperability between any vendor's TCP/IP protocol layer and applications programs 30. The next element stored in memory system 28 is Transmission Control Protocol/Internet Protocol ("TCP/IP") protocol layer 34. This element performs a large number of functions including packetizing data by attaching a header and footer to a block of data to be transmitted over a network and setting up a two-way connection between server 8 and client 20. A relatively small slice of bandwidth is allocated for communication upstream from the client to the server and a large amount of bandwidth is allocated for transmissions in the other direction.

See *Beighe* 2:46-62.

Argument

Despite *Nair's* localization of the buffers in the "buffer pool" that may be used by various protocol modules, the Examiner asserts that *Nair* "discloses allocating a system supplied buffer to the server application (Page 3 [0025].)" *Final Office Action*, p.3. Respectfully, Applicants disagree. The material at paragraph 25 from *Nair* cited by the Examiner discusses the allocation of a buffer by the protocol module, <u>when it receives a data packet ultimately delivered to an application</u>. On this point, *Nair* provides, "[t]he process diagrammed in FIG. 2 contemplates receiving a data frame at a machine in which an embodiment of the present invention is implemented." *Nair*, ¶ 23.

The buffers included in the "buffer pool" disclosed by *Nair* are used exclusively and independently by the "protocol modules" in processing data from, or passing data to, an application program. Thus, as taught by *Nair*, any buffers used by the server application are separate from any buffers used by a protocol module from a "buffer pool." Applicants point out the shortcoming of this conventional approach by noting:

> synchronous and asynchronous I/O both suffer from various problems. In the case of synchronous processing, the application is blocked/idle until the data is successfully sent to the client. In addition, the application's buffer is unusable during this time. In contrast, asynchronous processing is advantageous because control is returned to the application more quickly. However, asynchronous processing sufferers from the over

> utilization of storage because the application buffers and system buffers are needed. In addition, some additional system overhead is generated by virtue of the data copy from the application buffer to the system buffer.

Application, ¶ 11.

Because Nair is directed to the use of a localized buffer pool used exclusively by the protocol modules, *Nair* fails to disclose a system supplied buffer being allocated to a server application. In fact, *Nair* discloses that once the data frame is provided to the server application "**the buffer [used by the network protocol software modules] is no longer needed.**" Clearly, the operations performed by the server application are distinct from those used to manage a buffer within different layers of the protocol stack. The present claims, however, are directed to processing that occurs *after* data has been processed through a protocol communications stack, i.e., after the data is, in the words of *Nair*, "provided to an application software program." Thus, Applicants submit that *Nair* fails to disclose allocating a system-supplied buffer to the server application in response to a request from a server application.

Nair does disclose that a protocol module may allocate a buffer for data transmissions "down" the protocol stack. See Nair, ¶ 23, 30. The Examiner relies on this teaching to assert "*Nair* additionally describes in paragraph [0030] (page 3) that allocation occurs in the same manner when data is being transmitted by a server application." See Advisory Action, p.3. However, nothing from this passage describes the operations of the server application, other than supplying a "data frame" to the protocol modules "for processing." Quite the contrary, as disclosed in *Nair*, the operations of the server application are a "black box" to the protocol modules. For example, Figure 2 does not even illustrate the "higher level application;" instead, this figure simply includes an arrow leading from the TCP module 112. It is not until <u>after</u> receiving a data frame, either from the network connection or form a "higher level application," that *Nair* discloses doing anything at all – *Nair* is silent on what occurs before the data frame is received, which is where the allocation suggested by the Examiner (Advisory Action, p.3) would occur.

Applicants submit, therefore, that *Nair* fails to disclose the operations of allocating a system supplied buffer to a server application, and instead, that *Nair*

discloses the use of a private, localized buffer pool from which to allocate a buffer after receiving a data frame. Accordingly, Applicants believe claims 1, 12, and 24 are allowable over *Nair* in view of *Beighe*.

Regarding claims 2, 3, 5-10,13,15-21 and 25-31, each of these claims depends from one of claims 1, 12, and, 24. As Applicants believe the above remarks demonstrate that *Nair* in view of *Beighe* fails to disclose each and every limitation of the independent claims, Applicants believe that these dependent claims are allowable.

CONCLUSION

The Examiner errs in finding that claims 1-3, 5-10,12-13,15-21 and 24-31 are unpatentable over *Nair* in view of *Beighe* under 35 U.S.C. § 103(a). Withdrawal of the rejection and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan
Registration No. 44,227
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellants

CLAIMS APPENDIX

1. (Previously Presented) A method of processing messages in a computer, comprising:

in response to a request from a server application, allocating a system-supplied buffer to the server application, wherein the server application is configured to exchange data with a client application running on another computer using a network-based socket, and wherein the system supplied buffer is of a sufficient size to contain the data;

writing the data to the system-supplied buffer;

passing the system-supplied buffer to the network-based socket to allow the server application to continue processing while the data is sent to the client; and

sending, by way of the network-based socket, the data from the system-supplied buffer to the other computer via a network; and

freeing memory consumed by the system supplied buffer.

2. (Original) The method of claim 1, wherein the messages are client-server messages.

3. (Original) The method of claim 1, wherein the data is sent over a sockets streaming protocol.

4. (Canceled)

5. (Original) The method of claim 1, wherein sending is performed without first copying the data into another buffer.

6. (Previously Presented) The method of claim 1, wherein the writing is performed by the server application.

7. (Previously Presented) The method of claim 1, further comprising, prior to providing the system-supplied buffer to the server application:

receiving, by a socket, other data from the another computer via the network; and

allocating the system-supplied buffer to contain the other data.

8. (Previously Presented) The method of claim 1, wherein providing the system-supplied buffer to the server application comprises acquiring, by a socket, the system-supplied buffer from memory space not allocated to the server application.

9. (Previously Presented) The method of claim 1, wherein the system-supplied buffer is provided to the server application by a socket in response to a buffer acquisition function call from the server application.

10. (Previously Presented) The method of claim 1, wherein the system-supplied buffer is provided to the server application by a socket after the sockets server application requests client data received over a client connection with the another computer.

11. (Canceled)

12. (Previously Presented) A computer readable medium containing a sockets-based communications program which, when executed by a computer, performs operations for processing messages, the operations comprising:

in response to a request from a server application, allocating a system-supplied buffer to the server application, wherein the server application is configured to exchange data with a client application running on another computer using the communications program, and wherein the system supplied buffer is of a sufficient size to contain the data;

receiving the system-supplied buffer from the server application, wherein the system-supplied buffer contains data written to the system-supplied buffer by the server application;

sending, by way of the communications program, the data from the system-supplied buffer to the another computer via a network, thereby allowing the server application to continue processing while the data is sent to the client; and

returning the allocated system supplied buffer to the computer.

13. (Original) The computer readable medium of claim 12, wherein the messages are client-server messages.

14. (Canceled)

15. (Original) The computer readable medium of claim 12, wherein sending is performed without first copying the data into another buffer.

16. (Previously Presented) The computer readable medium of claim 12, wherein the writing is performed by the server application.

17. (Previously Presented) The computer readable medium of claim 12, further comprising, prior to allocating the system-supplied buffer to the server application:

receiving, by the communications program, over a socket, other data from the another computer via the network; and

allocating the system-supplied buffer to contain the other data.

18. (Previously Presented) The computer readable medium of claim 12, wherein providing the system-supplied buffer to the server application comprises acquiring, by a socket, the system-supplied buffer from memory space not owned by the server application.

19. (Previously Presented) The computer readable medium of claim 12, wherein the system-supplied buffer is provided to the server application by the communication program using a socket in response to a buffer acquisition function call from the server application.

20. (Previously Presented) The computer readable medium of claim 12, wherein the system-supplied buffer is provided to the server application by a socket configured by a receive operation issued from the server application and wherein the system-supplied buffer contains client data from the another computer.

21. (Original) The computer readable medium of claim 20, wherein providing the system-supplied buffer comprises allocating the system-supplied buffer according to a size of the client data.

22. (Original) The computer readable medium of claim 20, wherein the receive operation is configured with a buffer mode parameter indicating to the socket a buffer acquisition method for acquiring system-supplied buffer.

23. (Original) The computer readable medium of claim 22, wherein the receive operation is further configured with a record definition specifying to the socket a format of the client data.

24. (Previously Presented) A computer in a distributed environment, comprising:

a network interface configured to support a network connection with at least one other computer in the distributed environment;

a memory containing contents comprising:

an operating system;

a server application;

a sockets-based communication facility;

a system-owned memory space from which to allocate system-supplied buffers;

an application-owned memory space owned by the server application; and

a processor configured by at least a portion of the contents to perform operations for processing client-server messages, the operations comprising:

in response to a request from the server application, allocating a system-supplied buffer to the server application, wherein the server application is configured to exchange data with a client application running on another computer using a network-based socket, and wherein the system supplied buffer is of a sufficient size to contain the data.

25. (Original) The computer of claim 24, wherein the distributed environment is a client-server environment.

26. (Previously Presented) The computer of claim 24, wherein a protocol stack is configured for a sockets streaming protocol.

27. (Original) The computer of claim 24, wherein the processor is configured to send the data without first copying the data into another buffer.

28. (Previously Presented) The computer of claim 24, wherein providing the system-supplied buffer to the server application comprises acquiring, by the socket, the system-supplied buffer from the system-owned memory space.

29. (Previously Presented) The computer of claim 24, wherein the operations performed by the processor further comprise:

writing data into the system-supplied buffer;

returning the system-supplied buffer to the socket-based communication facility; and

sending the data from the system-supplied buffer to the at least one other computer.

30. (Previously Presented) The computer of claim 29, wherein the system-supplied buffer is returned to the socket-based communication facility on a send operation and wherein sending comprises detaching the system-supplied buffer from the send operation to allow the server application to continue processing while the data is sent.

31. (Previously Presented) The computer of claim 24, wherein the processor is configured to provide the system-supplied buffer to the server application by the socket in response to a buffer acquisition function call from the server application.

32. (Previously Presented) The computer of claim 24, wherein the socket is configured by a receive operation issued from the server application and configured with a buffer mode parameter indicating to the socket a buffer acquisition method for acquiring system-supplied buffer and wherein the system-supplied buffer contains client data from the at least one other computer.

33. (Original) The computer of claim 32, wherein providing the system-supplied buffer comprises allocating the system-supplied buffer according to a size of the client data.

34. (Original) The computer of claim 32, wherein the receive operation is further configured with a record definition specifying to the socket a format of the client data.